VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Optimalizace nářezových plánů
Optimization of Cutting Schemes

2013 **Martin Zonyga**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Martin Zonyga**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Optimalizace nářezových plánů
Optimization of Cutting Schemes

Zásady pro vypracování:

Nářezové plány jsou určeny k přípravě dřevěných výrobků. Pokud je plán navržen správně, minimalizuje se množství odpadu, který již nelze užít k další výrobě a tím pádem i cena, kterou společnost musí za materiál platit. Ve velkovýrobách může jít o statisícové až miliónové částky. Z těchto důvodů je zajímavá optimalizace nářezových plánů. Cílem této práce prostudovat možnosti optimalizací nářezových plánů a implementovat některou z vybraných metod.

Úkoly:
1. Nastudovat metody optimalizace nářezových plánů.
2. Implementovat vybranou metodu.
3. Prakticky ověřit výsledky na objemu odpadu před a po optimalizaci.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Karel Mozdřeň**

Datum zadání: 16.11.2012

Datum odevzdání: 07.05.2013

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 07. května 2013

Zonyga

.....

Martin Zonyga

V souvislosti s vypracováním bakalářské práce bych rád poděkoval za odbornou a profesionální podporu Ing. Karlu Mozdřeňovi, vedoucímu bakalářské práce.

Abstrakt

Nářezové plány slouží jako přesné návody při řezání plošných dílů. Vhodné uspořádání dílů je nezbytné pro zmenšení odpadu při řezání a zrychlení práce na pile. Tím docílíme snížení vstupních nákladů na materiál a možnosti zvládnout více zakázek za kratší čas, což povede k lepší konkurenceschopnosti. V této práci budeme navrhovat nářezové plány pro dřevěné výrobky, které se budou řezat na pile s předřezem. Tyto nářezové plány budeme optimalizovat vzhledem k množství odpadu a jednoduchosti při řezání.

Klíčová slova

Nářezové plány, C#, optimalizace, SVG, XML

Abstract

Cutting schemes are used as guides for precise cutting of planar parts. Suitable arrangement of parts is necessary to reduce the waste in cutting and acceleration of work on the sawmill. This we will achieve reducing input costs of materials and options to handle more work in less time, leading to improved competitiveness. In this work we will design cutting plans for wood products that will be cut at the sawmill with scoring unit. These cutting plans will be optimized with considering to the amount of waste and simplicity of cutting.

Key Words

Cutting schemes, C#, optimization, SVG, XML

Seznam použitých symbolů a zkratek

LDTD – Laminované dřevotřískové desky
W3C – The World Wide Web Consortium
XML – Extensible Markup Language
DOM – Document Object Model
XSL – eXtensible Stylesheet Language
SVG – Scalable Vector Graphics
DTD – Document Type Definition

Obsah

Seznam obrázků

1. Úvod

Pracuji ve stolařské firmě, která vyrábí nábytek a vybavení především pro školní a kancelářské potřeby. Tento nábytek je vyráběn především z laminovaných dřevotřískových desek (LDTD), o kterých ještě bude zmínka níže. Tyto LDTD se prodávají jako plošný materiál, který je nutno dále rozřezat na menší díly pro další zpracování. Rozřezané díly se dále olepí, vyvrtají a následně se z nich poskládá nový nábytek.

Když firma získá novou objednávku na nábytek, vedoucí dřevo-výroby musí zajistit a objednat materiál na výrobu a vytvořit nářezové plány. Tyto nářezové plány se vytváří z ploten LDTD, kdy je na tuto plotnu nutno poskládat mnoho menších dílů. Pokud je plán navržen správně, minimalizuje se množství odpadu, který již nelze užít k další výrobě a tím pádem i cena, kterou společnost musí za materiál platit. A právě na výrobu těchto nářezových plánů se v této bakalářské práci zaměřím.

1.1. Laminované dřevotřískové desky

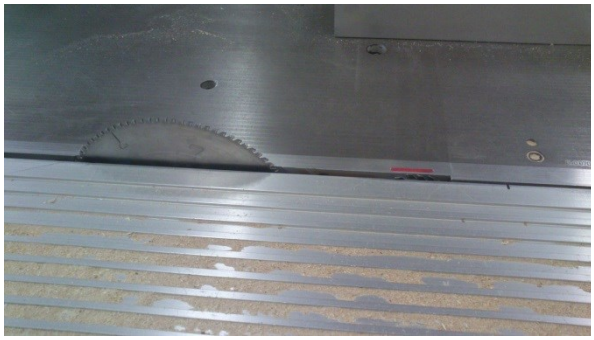
V této společnosti se v drtivé většině případů nábytek vyrábí z LDTD. Dřevotřískové desky [1] se vyrábí slisováním dřevěných třísek a pojiva. Jejich povrch je přírodní a můžeme v něm rozeznat drobnou kresbu a barvu rozemleté dřevěné drti, respektive všesměrně uložených třísek, zpravidla ve třech vrstvách. Jedná se o velkoplošný materiál lisovaný z třísek jehličnatých a listnatých dřevin, které jsou spojeny nezávadnou formaldehydovou pryskyřicí. Dodávají se s rovnou hranou nebo s drážkou. DTD se prodávají v surovém stavu, existují rovněž DTD laminované (s nalisovaným dekorativním papírem v dekoru dřeviny nebo v uni barvě) a DTD dýhované (povrchově upravené přírodní dýhou různých dřevin). Povrchová úprava vzniká nalisováním dekorativního papíru impregnovaného aminoplastickými pryskyřicemi. Povrchová vrstva laminátu s dekorativním povrchem a definovanou strukturou povrchu je odolná vůči krátkodobému působení vody, zvýšené teplotě a chemikáliím používaným v domácnosti. Povrch lamina je snadno omyvatelný a bez zápachu. Široká paleta dekorů lamina a výběr z mnoha struktur povrchu nabízí využití v mnoha odvětvích nábytkářského průmyslu.

1.2. Vybavení dílny

1.2.1. Formátovací pila

V dílně je velká stojatá formátovací pila. Pila má hlavní kotouč o průměru 300 mm a ve vzdálenosti 15 cm před ním je kotouč předřezový (viz Obrázek 1). Hlavní kotouč se točí po směru hodinových ručiček, předřezový kotouč opačným směrem. Směr točení zajišťuje, že zuby kotouče se „zakusují“ do materiálu a díky tomu je řez z vrchní strany čistý a není vyštípaný. Na spodní straně řezaného materiálu je předřezový kotouč, který bývá o 0,3 až 0,5 mm širší a točí se opačným směrem. Úkolem předřezového kotouče je zajistit prořízntutí pouze malé vrstvy materiálu ze spodní strany.

Zbytek tloušťky řeže hlavní kotouč, díky tomu je celý řez čistý a bez vad. Pila má také pojezdový vozík pro snadnější manipulaci s plotnami během řezání a spodní i horní odsávání prachových částic.



Obrázek 1: Hlavní a předřezový kotouč

Obrázek 2: Formátovací pila

1.2.2. Olepovačka hran

Poté co byly díly přesně nařezány, mohou se řezané hrany olepit. K tomuto účelu u nás slouží Olepovačka hran OTT TORNADO. Tato olepovačka je velmi univerzální a všestranný stroj, který používá lepidlo CombiMelt. Ovládá se pomocí dotykové obrazovky a PC a je plně automatizovaná. Má vlastní frézy pro předfrézování 0,1 až 3 mm, díky kterému se odstraní nerovnosti plochy způsobené předřezem. Tloušťka olepovaného materiálu může být 8 až 52 mm. Tloušťka hrany může být 0,4 až 3 mm. Rychlost posuvu, včetně zaoblením rohů, je až 14 m / min.

Obrázek 3: Ovládací obrazovka olepovačky hran

Obrázek 4: Olepovačka hran

Po olepení se díly dále zpracovávají až ke konečnému složení výrobku, zabalení a následné expedici. Tato část vybavení dílny však už nemá vliv na tvorbu nářezových plánů.

2. Současný stav

Nářezový plán je přesné rozmístění a počet dílů na plotně, podle kterého se má plotna nařezat. Příkladem může být Obrázek 5. Pokud je plán navržen správně [2], minimalizuje se množství odpadu, který již nelze užít k další výrobě a tím pádem i cena, kterou společnost musí za materiál platit. Ve velkovýrobách může jít o statisícové až miliónové částky.

2.1. Stávající nářezové plány

Stávající nářezové plány jsou vytvářeny v programu Merick Calc WF, který je popsán v bodě 3.1 Merick Calc WF. Navíc v základní verzi neumí přímo ukládat výrobky a jejich díly pro jejich snadnou práci při příštím použití. Při každé nové zakázce je nutné nově ručně vytvořit seznam dílů a jejich rozměrů, žádná jednoduchá možnost načtení výrobků a jejich dílů zde není. Kvůli těmto problémům má šéf dřevovýroby všechny výrobky a jejich díly psané v MS Excelu a při ručním přepisování a kopírování do prostředí Mericku se stávalo, že docházelo k překlepům a chybám. Dalším neduhem byla nemožnost hromadné změny, např. když se měnila stará olepovačka za novou. Nová uměla funkci předfrézování před lepením, což mělo za následek, že se řezané rozměry všechny dílů, které se olepovaly, musely změnit.

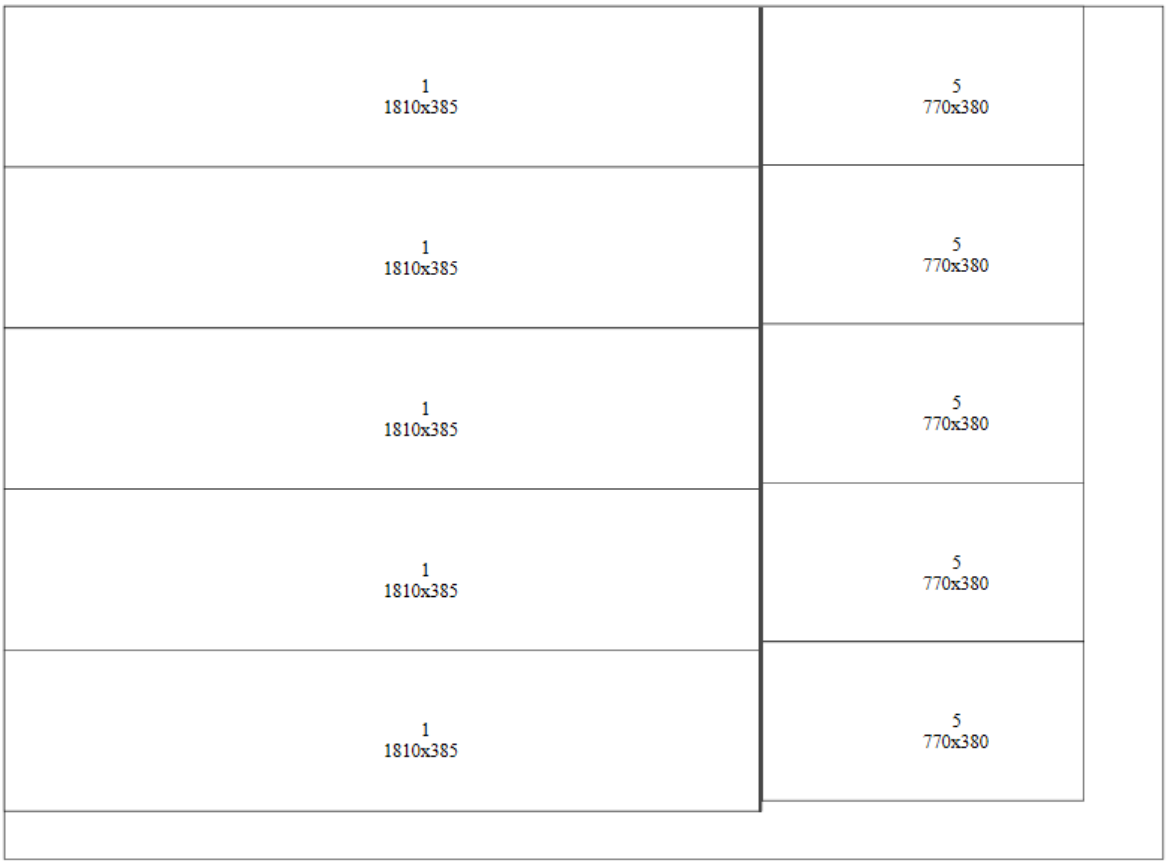
To byly důvody, proč jsem se rozhodl pro toto téma bakalářské práce. Můj program budu pro firmu vyvíjet v rámci bakalářské práce a měl by přesně pasovat do tohoto prostředí. Program bude mít evidenci výrobků a kusů, ze kterých se tyto výrobky skládají. Tím odpadne část, kdy se musely data přepisovat mezi různými systémy a eliminuje se tak mnoho chyb. Také výroba nářezových plánů bude rychlejší a pohodlnější. Program bude mít také návaznost na olepovačku, což umožní jednoduché změny v případě poruchy nebo výměny. U stávajících nářezových plánů se také občas stávalo, že byla dávka několika stejných dílů vedle sebe a mezi nimi se objevil díl jiný se stejnými nebo podobnými rozměry. Toho bych se také v této práci rád vyvaroval.

2.2. Zakrácení na délku vs. na šířku

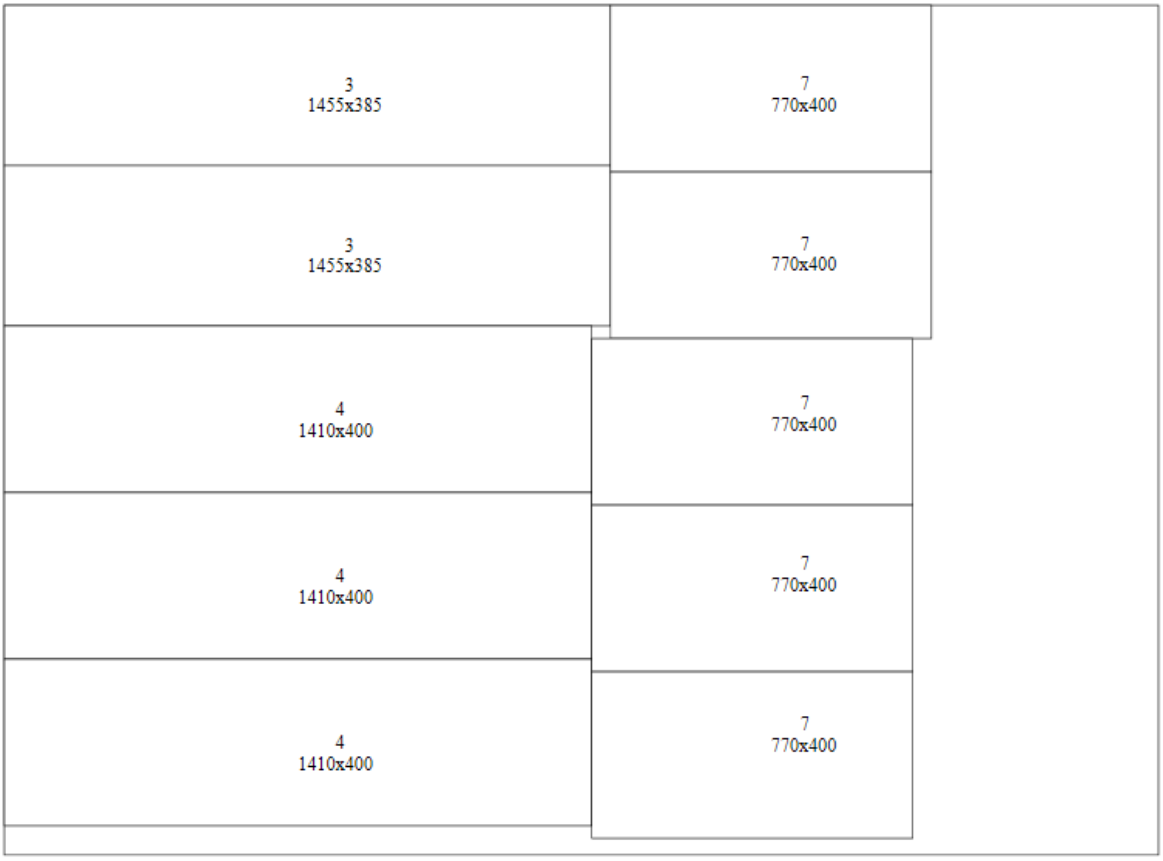
Díky předřezu má takovéto řezání velmi specifické požadavky na výrobu nářezových plánů. Není možné použít nějaký jednoduchý algoritmus (např. algoritmy popsané v bode 4.4) a naskládat jednotlivé kusy na plotnu tak, aby zabraly co nejméně místa. Tímto způsobem by nám vznikaly i nářezové plány jako je například Obrázek 6: Příklad špatného nářezového plánu, který nejsme schopni nijak na této pile nařezat. Řez po délce i po šířce by znamenal, že předřezový kotouč ze spodní strany nařízne i další díly, čímž by je znehodnotil.

Z toho plyne, že pokud chceme provést nějaký řez, musí tento řez být veden po celé délce, popřípadě šířce plotny. Není tedy možné poskládat „zubaté“ nářezové plány jako je ten na Obrázek 6: Příklad špatného nářezového plánu. Při tvorbě nářezových plánů tedy máme na výběr ze dvou výchozích stavů a to skládat díly podle délky nebo podle šířky. Při skládání podle šířky mohou vznikat zbytky, které se později lépe upotřebí u jiných zakázek nebo malých dílů. Ale protože řezání je poměrně časově náročné, zvolil jsem rychlejší metodu a tou je řezání podle délky. Tj. nejdříve se uřeže pás na přesnou délku a až poté se rozřezává na přesnou šířku. Tím se obvykle ušetří několik řezů a práce je rychlejší a proto také levnější. Konkrétní příklad si můžeme vyzkoušet například na nářezovém plánu podobnému Obrázek 5.

Při řezání po šířce by nářezový plán neměl stejně dlouhé kusy skládané pod sebe, ale stejně široké kusy skládané za sebe. Při řezání na pásy by bylo nutné provést 6 řezů, čímž bychom dostali 5 pásů, a z nich každý by se musel ještě 3x řezat, abychom vyrobili díly zakrácené i na délku. To je celkem 21 řezů. Pro menší odpad a snížení počtu řezů bychom si mohli např. nahrubo zakrátit plotnu na délku (řez nejvíce vpravo). Tímto řešením bychom měli už 2 použitelné odřezky a počet řezů by se snížil na 17. Pokud ale provedeme nejdříve zakrácení pásů na délku, a až následné rozřezání na šířku, počet řezů klesne na 15. Nejlépe tedy vychází nařezat si nejprve pásy na přesnou délku a následně pásy rozřezat na přesnou šířku.



Obrázek 5: Příklad nářezového plánu



Obrázek 6: Příklad špatného nářezového plánu

3. Alternativní systémy

Tvorbou nářezových plánů se zabývá několik programů. Každý z těchto programů má svá specifika, ale v základu jsou vždy výsledkem více či méně zdařilé nářezové plány pro danou zakázku. Z těchto programů jsem vybral tři zástupce, které jsem popsal níže.

3.1. Merick Calc WF

Tento program [3] je určen pro vytváření nářezových plánů ze svitkových a deskových materiálů, hlavně pak okenních fólií. Kromě toho dokáže vytvořit nabídku pro zákazníka, evidovat více rozměrů materiálu a recyklovat zbytky materiálu. Program je určen pro operační systémy Windos95/98/ME/2000/XP/Vista/7. Není nijak náročný na výkon počítače, stačí Pentium 200MHz a lepší a zobrazení s rozlišením 800x600. Program svým vzhledem a ovládáním připomíná klasickou webovou stránku. Tento program se momentálně u nás v práci používá.

3.2. OPTIMIK

Program Optimik [4] slouží k automatickému návrhu nářezových plánů pro výrobce nábytku a ostatních výrobků z plošných materiálů (dřevo, plech, sklo…), jejichž výrobní sortiment je široký nebo neustále se měnící (zakázková výroba apod.) Mimo této základní úlohy dokáže automaticky vést a upravovat podrobnou evidenci materiálu a jednotlivých formátů, evidovat sestavy a vkládat je do zakázek.

Tento produkt je dle mého názoru ideálním komplexním řešením pro menší podnikatele. Má totiž hned několik modulů jako jsou Sklad, Obchodní partneři, Doklady, Výrobky, Zakázky a Nářezové plány. Pro potřeby oddělení dřevovýroby jsou některé tyto moduly zbytečné, protože o prodej se stará Obchodní oddělení. Program Optimik existuje i ve free verzi, která je ovšem oproti placeným verzím značně omezena. Nejlevnější placenou verzí je verze Hobby a nejdražší verzí je verze CNC. Tato verze už je určena pro CNC pily, pro které umí nářezové plány i exportovat.

3.3. Nowy Rozkroj

Nowy Rozkrój [5] je počítačový program pro optimalizaci nářezových plánů nábytkářských tabulí, skla a jiných plošných materiálů. Jednoduchá obsluha, všestranné použití a vysoká efektivita práce v programu vám pomohou ušetřit spoustu času, práce a materiálů. Jednou ze základních předností programu Nowy Rozkrój je schopnost spolupráce s programem pro projektování nábytku a navrhování interiérů PRO100. Spolupráce mezi těmito dvěma programy spočívá v automatickém exportu seznamu přířezů z programu PRO100 do programu Nowy Rozkrój. Díky této operaci nedochází k chybám, které se často vyskytují při ručním vypisování přířezů a lze významným způsobem snížit objem práce vynakládané při výrobě přířezů navržených v programu PRO100.

Program je vhodný například pro firmy, které fungují jako subdodavatelé. Firmy, které pouze nařežou, popřípadě nařežou a olepí a takto zpracované kusy předávají zákazníkovi. Program totiž mimo jiné umožňuje tisknout etikety (štítky) pro jednotlivé díly. Takto poznačené díly je možné dát zákazníkovy se seznamem dílů, kde je jejich popis.

Zde jsou uvedeny aktuální ceny jednotlivých programů ke dni 28. 04. 2013:

Merick Calc WF	93,35 € + DPH a poštovné
OPTIMIK 3.28 Hobby	82,80 € + 10,80 € podpora na 1 rok
OPTIMIK 3.28 CNC	718,80 € + 94,80 € podpora na 1 rok
Nowy Rozkroj verze 6.2.5	214,80 €

4. Návrh řešení

Pro výrobu nářezových plánů budu potřebovat zakázku. Zakázka bude uložena v XML souboru, jehož výhody jsou popsány níže. Z této zakázky si vezmu velikosti a počty dílů, které potřebuji umístit na plotny. Poté, co si připravím seznam dílů, mohu je začít skládat na plotnu. Začnu vždy tím největším dílem a ten umístím na začátek plotny. Následně se podívám jestli mám stejný díl vícekrát a pokud to bude možné umístím jej pod první díl. Když budu mít hotovou první řadu dílů, začnu vedle skládat druhou řadu stejným způsobem. Takto naskládám vedle sebe tolik řad, kolik bude možné. Jakmile se další díly na plotnu už nedají umístit, vezmu novou plotnu a celý proces opakuji. Toto provádím do té doby, dokud neumístím všechny díly a pak můžu výsledné plotny vykreslit.

4.1. XML

Pro ukládání svých výrobků a zakázek jsem zvolil formát XML. XML [6], neboli česky rozšiřitelný značkovací jazyk, je jazyk vytvořený pro přenášení a uchovávání dat. Je jakýmsi mezistupněm mezi binárním uložením dat (dobře čitelná pro počítač) a textovým uložením dat (dobře čitelná pro člověka). XML je značkovacím jazykem, ale oproti např. HTML byl navržen k přenášení dat a ne jejich zobrazení. Struktura dat je samo popisná, protože v dokumentu můžeme vytvářet vlastní tagy.

4.2. SVG

Pro výsledný grafický výstup jsem si vybral vektorový grafický formát SVG [7] (Scalable Vector Graphics). Výhodou vektorové grafiky je zejména možnost práce s objekty. Objekty mohou být čára, obdélník, kruh, mnohoúhelník nebo i text. Díky objektovému přístupu je možné měnit jednotlivé objekty (přesouvat, měnit barvy, velikosti, tvary) bez ohledu na ostatní objekty. Oproti tomu rastrová grafika má jakousi „mapu“ složenou z bodů, tzv. pixelů. Každý pixel má svou barvu a dohromady tvoří celý obraz. Z jednoho pixelu neurčíme nic, a protože pixelů je obvykle velké množství, takové obrazy bývají do objemu dat větší než stejné nebo podobné obrazy vytvořené vektorovou grafikou. Rastrová grafika se využívá například pro pořizování fotografií, což by vektorově pomocí křivek a matematiky by bylo náročné.

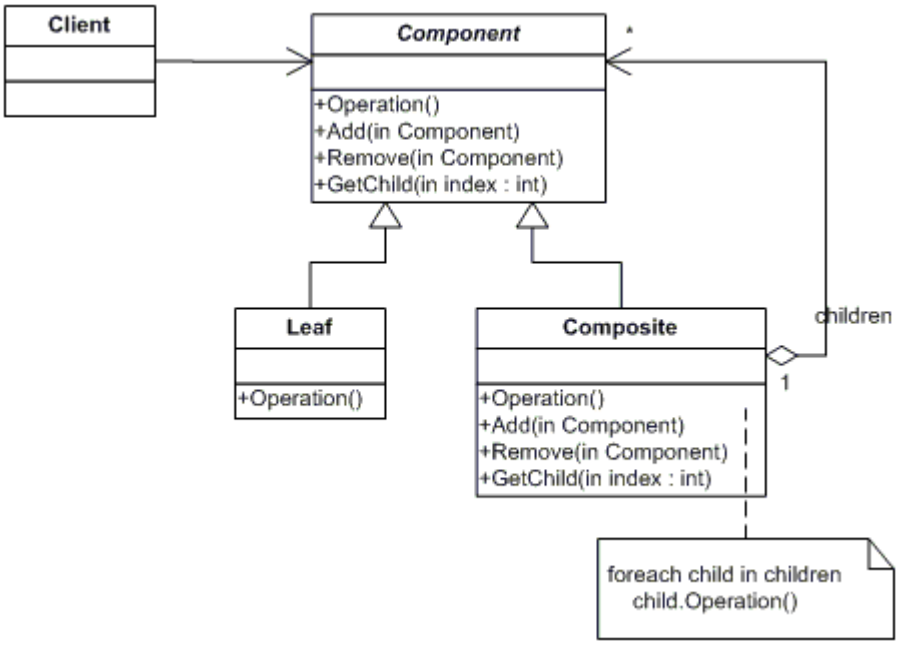
SVG formát je standard pro popis vektorové grafiky, je ve formátu XML, každý element a každý prvek může být animován, je součástí doporučení W3C, integruje s jinými standardy W3C, jako jsou DOM a XSL.
Hlavní přednosti SVG formátů jsou:

- SVG obrazy mohou být vytvořeny a editovány pomocí libovolného textového editoru
- SVG obrazy lze prohledávat, indexovat, scriptovat, a komprimovat
- SVG obrazy jsou škálovatelné
- SVG obrazy lze vytisknout ve vysoké kvalitě v jakémkoliv rozlišení
- SVG obrazy můžeme libovolně zoomovat bez ztráty kvality
- SVG je otevřený standard
- SVG soubory se skládají z čistého XML

Hlavním konkurentem SVG je Flash. Největší výhodou SVG oproti Flashi je dodržování jiných norem (jako např. XSL a DOM). Flash spoléhá na patentovanou technologii, která není open source.

4.3. Návrhový vzor Composite

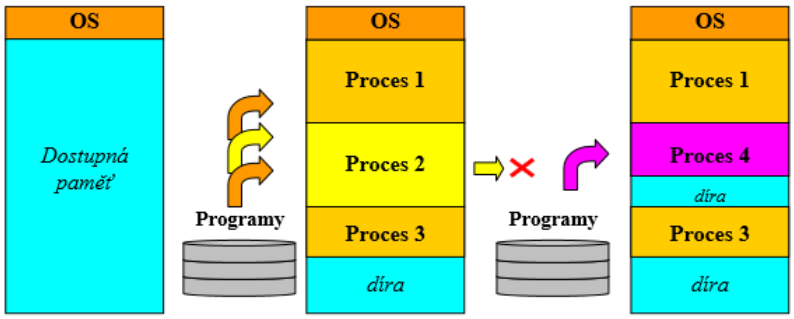
Vytvářím nářezové plány, které se vždy vztahují ke konkrétní zakázce. U této zakázky mám vždy několik ploten, které tvoří nářezové plány. Na každou plotnu umístím díly, které se na danou plotnu vlezou. Toto je vhodné pro použití návrhového vzoru Composite. Použití toho vzoru [8] je vhodné pro prvek (říkejme mu kontejner), který obsahuje prvky, z nichž každý může být kontejnerem. Struktura vzoru Composite je na Obrázek 7: Návrhový vzor Composite.



Obrázek 7: Návrhový vzor Composite

4.4. Algoritmy pro správu paměti

Při rozmisťování dílů na plochu bych mohl vycházet z algoritmů používaných pro práci s pamětí v PC. Velikost přidělované paměti [9] [10] se přizpůsobuje požadavkům právě zaváděného procesu. Správce takto může mnohem lépe s pamětí hospodařit. Na druhou stranu je metoda náchylná na vznik externí fragmentace jako důsledek neuspořádaného rušení a zavádění procesů. Jinými slovy, díra po ukončeném procesu se obvykle neshoduje s požadavky nového procesu, takže buď se díra nedá použít (nový proces je větší) nebo není využita celá (nový proces je menší, tím vzniká nová díra). Tady je vidět analogie s nářezovými plány. Pod dírou v paměti si můžeme představit volné místo na plotně a externí fragmentace představuje nevyužitelný odpad.



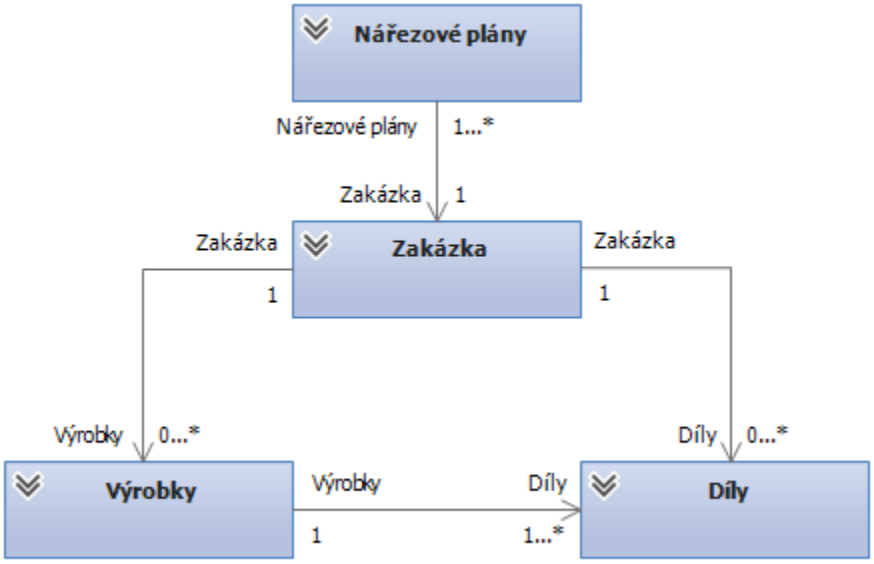
Obrázek 8: Metoda přidělování bloků proměnné velikosti a vznik externí fragmentace

K nalezení vhodného bloku volné paměti (díry) podle požadavků procesu se používají různé strategie výběru:

First Fit – při této strategii správce paměti prochází všechny díry, dokud nenalezne dostatečně velkou díru, jejíž velikost se shoduje nebo je větší, než je požadavek nového procesu. Z nevyužité paměti se stává nová, výrazně menší díra. Důsledkem je značná externí fragmentace.
Last Fit – tato strategie se od předchozí liší pouze tím, že se paměť prohledává odzadu. Obsadí se tudíž poslední díra dostatečné velikosti. To má za následek, že se nejprve spotřebovává dosud nepoužitá paměť bez vzniku fragmentace. Teprve až když se vyčerpá, správce začne hledat v místech dříve obsazené paměti.
Best Fit – princip této strategie je založen na přidělení paměti z volného prostoru, který se nejvíce přibližuje požadavkům procesu. Ve své podstatě se hledá nejmenší vyhovující díra. Tento postup převážně minimalizuje celkovou velikost fragmentované paměti. V určitých případech však může (oproti předchozím strategiím) také fragmentaci zvýšit. Nevýhoda metody spočívá v její složitosti, jež se projevuje celkově nižším výkonem. Jednoduše řečeno, správce musí pokaždé porovnat velikosti všech děr.
Worst Fit – je protikladem předchozí strategie. Procesu se přidělí paměť z největšího volného prostoru. Metoda předpokládá, že i po přidělení paměti zbude dostatečná část pro potřeby jiného procesu.

Tyto metody jsou vhodné pro práci, pokud mám jenom jeden rozhodující faktor jako je velikost paměti. V praxi lze tyto metody použít například při výběru a řezání tyčí nebo trubek, kdy jediným faktorem je délka. V případě nářezových plánů máme rozhodující faktory dva (délku a šířku) a proto budu muset tyto algoritmy modifikovat. Buď budu muset nářezové plány převést z 2D problému na problém pouze s jedním rozměrem nebo najít jiný vhodnější algoritmus.

4.5. Konceptuální model



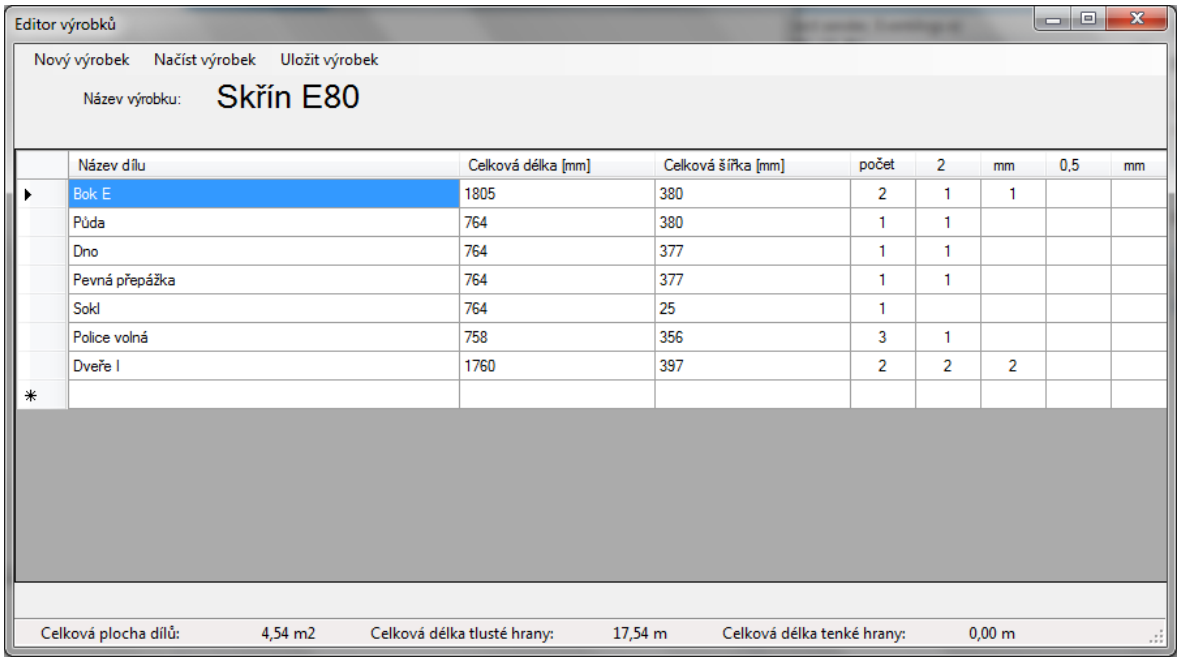
Obrázek 9: Konceptuální model nářezových plánů

Tento konceptuální model zobrazuje základní strukturu nářezových plánů, ze které budu při návrhu a tvorbě vycházet.
Nářezové plány tvoří cíl, ke kterému se chci dopracovat. Nářezové plány se vytváří pro konkrétní zakázku a ty se obvykle skládají z několika standartních výrobků (skříně, stoly, katedry), ke kterým může být přidáno ještě několik volných dílů (typicky desky lavic, které mají kovovou kostru). Výrobky se také skládají z dílů. Pro každou zakázku bývá obvykle několik nářezových plánů.

5. Popis tříd

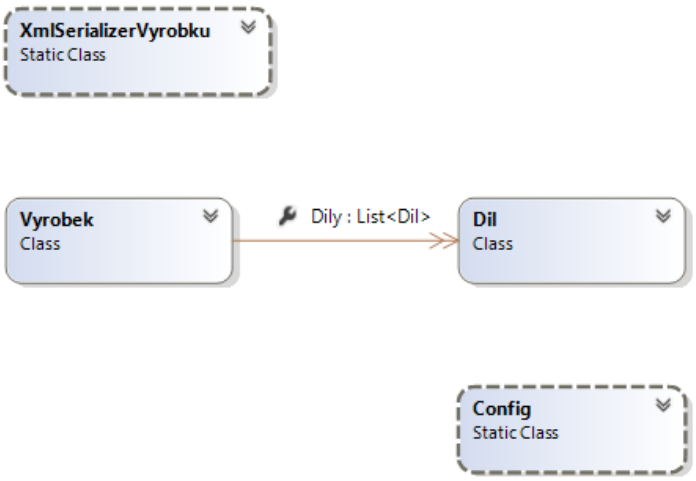
V této části se podíváme na jednotlivé třídy, jejich vlastnosti a chování. Projdeme si všechny 3 logické části projektu. Rozdělil jsem je na část pro výrobky, část pro zakázky a samotnou výrobu nářezových plánů.

5.1. Kusovník



Obrázek 10: Formulář pro vytváření a editaci výrobků

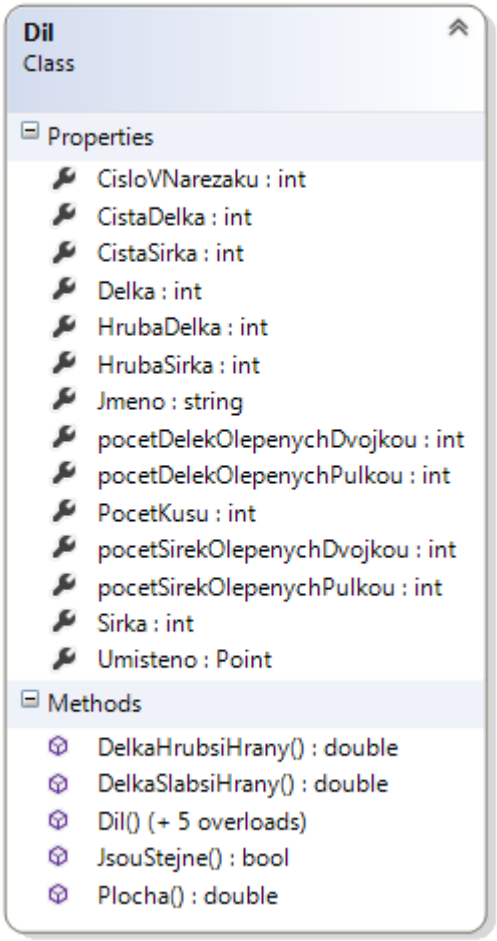
Obrázek 10 je formulář pro editaci výrobků. Každý výrobek se skládá z několika dílů. Pomocí tohoto formuláře je možné přidávat nové výrobky a editovat ty stávající. U dílů se zadávají konečné rozměry po olepení, čistý formát dílů k řezání si program dopočítá sám.



Obrázek 11: Třídní diagram kusovníku

Na Obrázek 11 je vidět třídní diagram pro část kusovníku. Výrobek obsahuje seznam dílů, z kterých je složen. Díl si bere data z třídy Config a o ukládání a načítání dílů se stará třída XmlSerializerVyrobku.

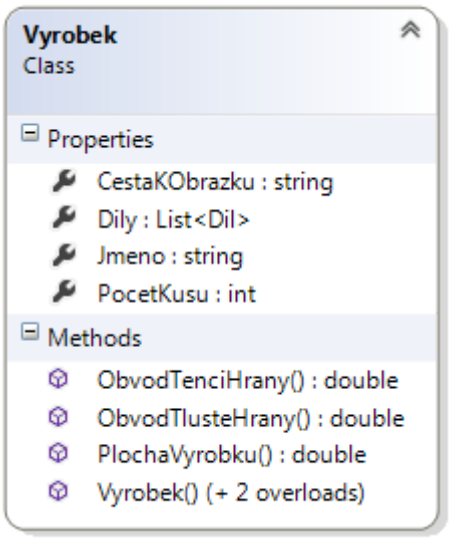
5.1.1. Třída Díl



Obrázek 12: Třída Díl

Tato třída reprezentuje díly. Každý díl musí mít povinně svou délku, šířku, jméno a počet olepených stran. Při vytváření nové instance lze zadat také počet kusů tohoto dílu a počet stran olepených slabší hranou. Hrubá délka a šířka se dopočítají podle hodnot zvolených v Konfiguračním souboru a jeho umístění a číslo v nářezovém plánu se vyplňují až při tvorbě nářezových plánů.
Čistá délka a šířka jsou skutečné rozměry, jaké výsledný díl musí po nařezání mít. K těmto rozměrům se připočte několik mm jako síla pilového kotouče popř. ke korekci drobných chyb při řezání a vzniknou hrubé rozměry, s kterými se počítá v nářezových plánech.
Každý díl umí spočítat svou plochu a délky hran, které jsou potřeba na jeho olepení. Obsahuje také statickou metodu, která porovnává 2 objekty, jestli jsou stejné. Třída Dil má atribut [Serializable] a bezparametrický konstruktor, aby ji bylo možné ukládat do XML.

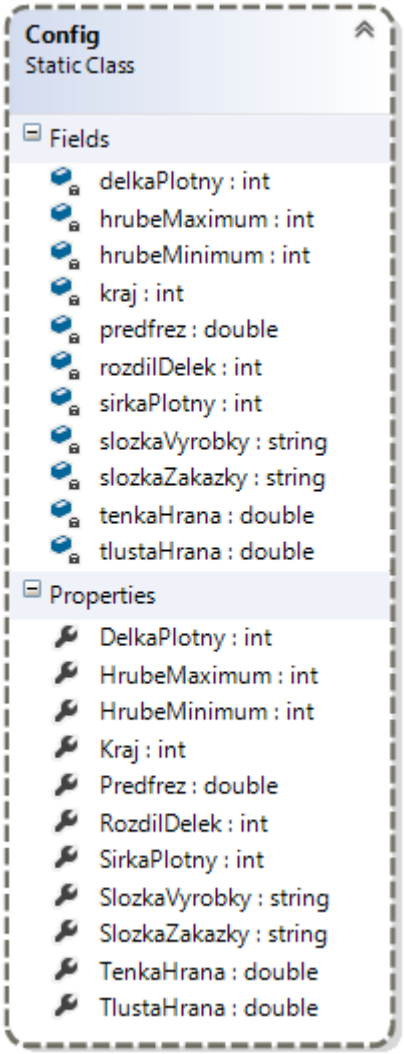
5.1.2. Třída Výrobek



Obrázek 13: Třída Výrobek

Třída reprezentující výrobek. Výrobkem může být skříň, stůl nebo katedra. Každý výrobek má své jméno a seznam dílů, ze kterých se skládá. Umí vypočítat kolik metrů hrany bude na jeho výrobu potřeba a jakou plochu zabere. Třída Vyrobek má atribut [Serializable] a bezparametrický konstruktor, aby ji bylo možné ukládat do XML.

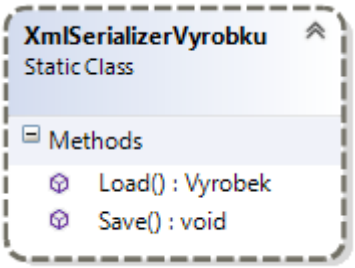
5.1.3. Třída Config



Obrázek 14: Třída Config

Třída Config představuje konfigurační soubor pro nastavení celého běhu programu. Je zde uvedeno jak velké plotny (tabule) se budou pro nářezové plány používat, jak velká je oblast kolem krajů, která se nebude využívat, kolik mm se bude přidávat ke každému dílu v nářezových plánech, jak velký rozdíl délek může být ve sloupci, tloušťky lepených hran a velikost předfrézu před lepením. Také je tu umístění složky pro výrobky a zakázky.

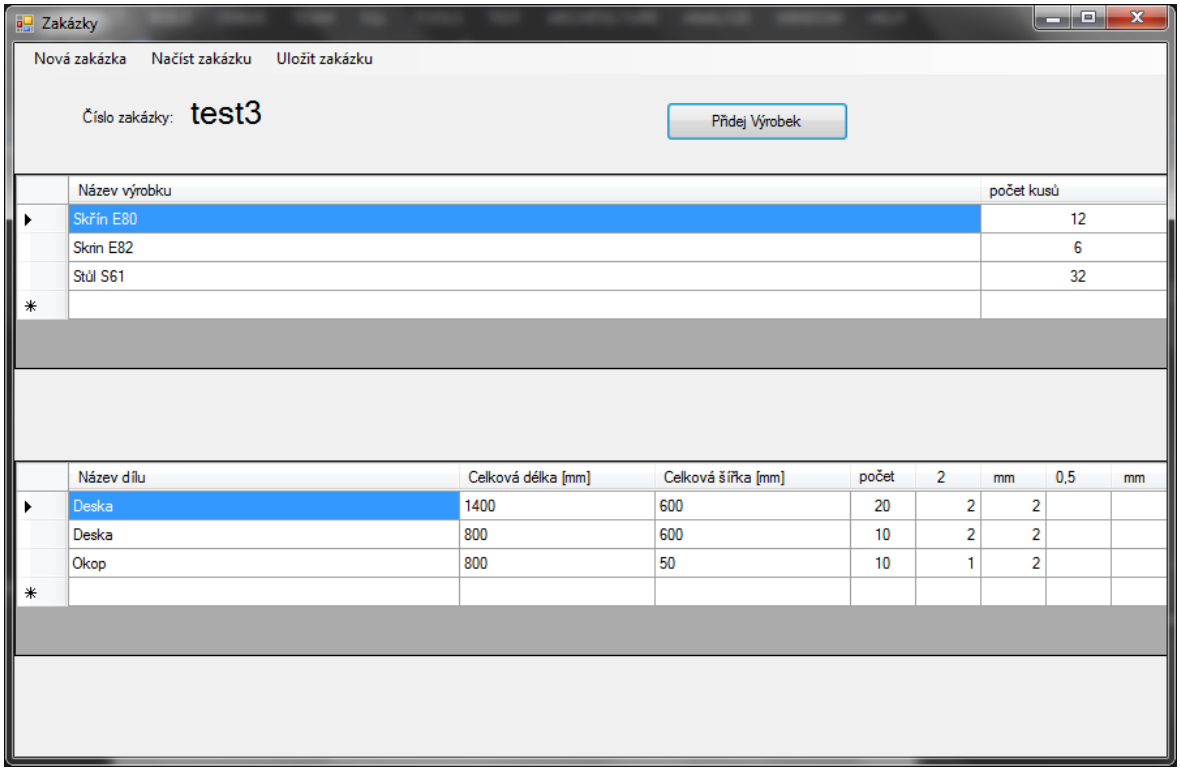
5.1.4. Třída XmlSerializerVyrobku



Obrázek 15: Třída XmlSerializerVyrobku

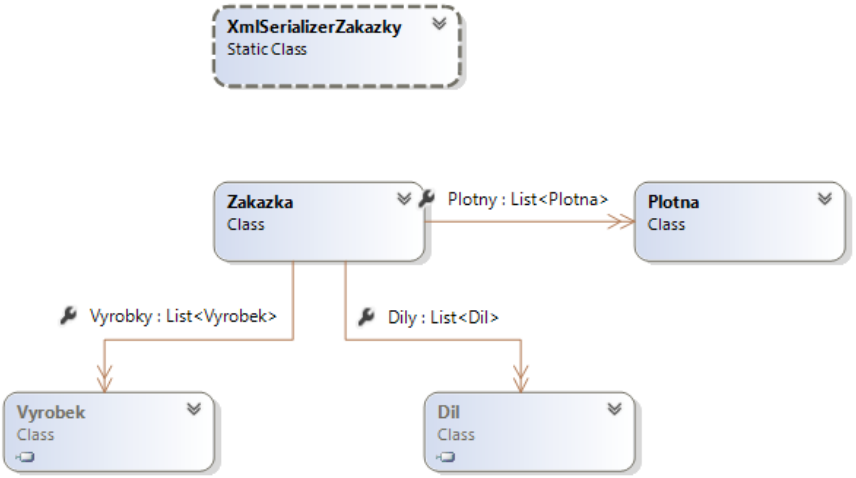
Toto je statická třída sloužící pro ukládání a načítání výrobků pomocí serializace do XML souborů.

5.2. Zakázky



Obrázek 16: Formulář pro vytváření a editaci zakázek

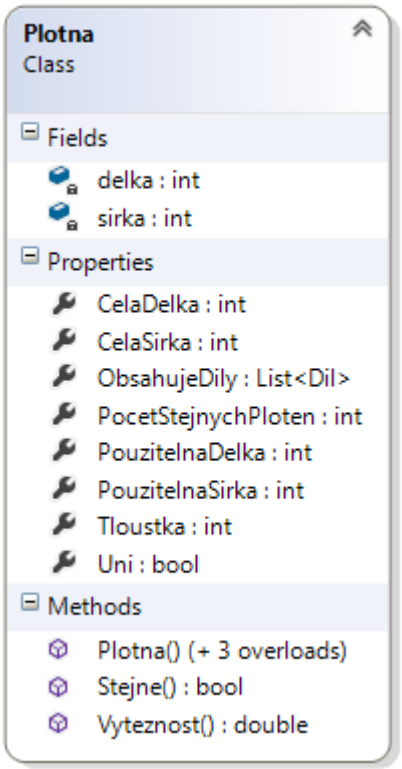
Obrázek 16 je formulář pro tvorbu zakázek. Zakázka se skládá z výrobků, které se přidávají do první tabulky nebo se v zakázce mohou objevit také volné desky, které nejsou součástí žádného výrobku. Zakázku opět můžeme ukládat i načítat.



Obrázek 17: Třídní diagram zakázek

Na Obrázek 17 je zobrazen třídní diagram pro zakázkovou část. Vidíme zde třídu Zakázka, která je složena ze seznamu výrobků a seznamu dílů. Taktéž obsahuje seznam ploten, podle kterých se budou nářezové plány vytvářet. Seznam ploten je prázdný do doby, než začnu tvořit nářezové plány. Poté bude každá plotna obsahovat seznam dílů, které k ní patří.

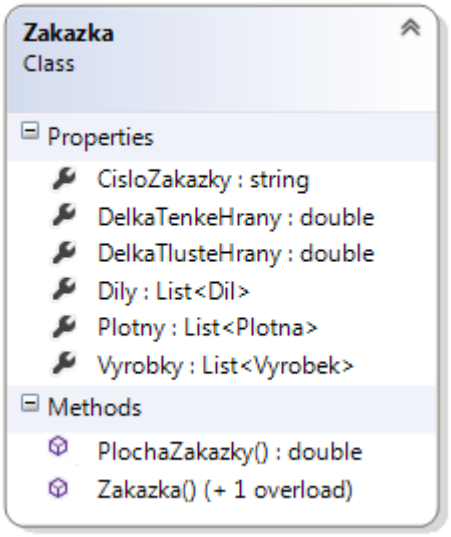
5.2.1. Třída Plotna



Obrázek 18: Třída Plotna

Třída reprezentující plotnu lamina. Plotna má délku, šířku a tloušťku. V již vytvořeném nářezovém plánu každá plotna obsahuje seznam dílů, které jsou na ní umístěny a počet ploten, které obsahují stejné díly a jsou stejně umístěné. Obsahuje také statickou metodu pro ověření, zdali není stejná s nějakou jinou plotnou a metodu pro výpočet výtěžnosti plotny, tj. využitou plochu ku ploše celkové.

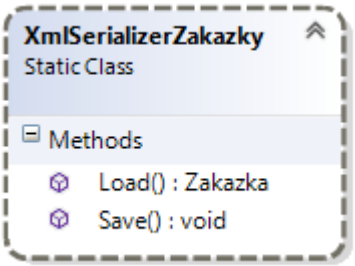
5.2.2. Třída zakázka



Obrázek 19: Třída Zakázka

Třída reprezentující zakázku. Každá zakázka má své číslo a obsahuje seznam výrobků, popřípadě dílů, které se mají v dané zakázce zpracovat. Z těchto informací můžeme spočítat délku hrany, která bude na zakázku potřeba a po výrobě nářezových plánů i počet ploten, které bude nutné pořezat.

5.2.3. Třída XmlSerializerZakazky



Obrázek 20: Třída XmlSerializerZakázky

Statická třída sloužící pro ukládání a načítání zakázek pomocí serializace do XML souborů.

6. Tvorba Nářezových plánů

V této části už se podíváme na samotnou implementaci a zdrojové kódy výroby nářezových plánů. Nářezové plány se tvoří vždy ke konkrétní zakázce. Bez zakázky, potažmo seznamu dílců, by nebylo z čeho nářezové plány tvořit.

Postup tvorby nářezových plánů je následující:

1. Vytvořím zakázku
2. Vyberu všechny díly, které budu umísťovat do nářezových plánů
3. Dílům přiřadím čísla, které se budou v nářezových plánech zobrazovat
4. Umístím díly na plotny
5. Vyrobené nářezové plány vykreslím

6.1. Vytvoření zakázky

Zakázku vytvořím pomocí formuláře popsaného v bodě 5.2 Zakázky. Po vytvoření a uložení zakázky ji načtu pomocí deserializace z XML pro následné zpracování. Na Obrázek 21 je příklad zakázky.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Zakazka xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <CisloZakazky>222</CisloZakazky>
    <DelkaTlusteHrany>0</DelkaTlusteHrany>
    <DelkaTenkeHrany>0</DelkaTenkeHrany>
  - <Vyrobky>
    - <Vyrobek>
        <Jmeno>Stůl S61</Jmeno>
      - <Dily>
        + <Dil>
        + <Dil>
        - <Dil>
            <PocetKusu>1</PocetKusu>
            <CisloVNarezaku>0</CisloVNarezaku>
            <Delka>1458</Delka>
            <Sirka>352</Sirka>
          - <Umisteno>
              <X>0</X>
              <Y>0</Y>
            </Umisteno>
            <Jmeno>Záda stolu</Jmeno>
            <pocetDelekOlepenychPulkou>0</pocetDelekOlepenychPulkou>
            <pocetSirekOlepenychPulkou>0</pocetSirekOlepenychPulkou>
            <pocetDelekOlepenychDvojkou>1</pocetDelekOlepenychDvojkou>
            <pocetSirekOlepenychDvojkou>0</pocetSirekOlepenychDvojkou>
          </Dil>
        </Dily>
        <PocetKusu>1</PocetKusu>
      </Vyrobek>
    </Vyrobky>
  - <Dily>
    - <Dil>
        <PocetKusu>30</PocetKusu>
        <CisloVNarezaku>0</CisloVNarezaku>
        <Delka>800</Delka>
        <Sirka>600</Sirka>
      - <Umisteno>
          <X>0</X>
          <Y>0</Y>
        </Umisteno>
        <Jmeno>deska</Jmeno>
        <pocetDelekOlepenychPulkou>0</pocetDelekOlepenychPulkou>
        <pocetSirekOlepenychPulkou>0</pocetSirekOlepenychPulkou>
        <pocetDelekOlepenychDvojkou>2</pocetDelekOlepenychDvojkou>
        <pocetSirekOlepenychDvojkou>2</pocetSirekOlepenychDvojkou>
      </Dil>
    </Dily>
    <Plotny/>
</Zakazka>
```

Obrázek 21: Příklad zakázky

6.2. Seskládání dílů

V zakázce obvykle bývá několik stejných výrobků a každý z nich má několik stejných dílů. Pro každý takovýto díl vytvořím samostatný objekt. Díky tomu každý díl mohu následně vložit na plotnu a přiřadit mu souřadnice, na kterých se má nacházet. Pokud má díl malé rozměry (je krátký nebo úzký), v nářezovém plánu jej vynechám. Takovéto drobné kusy se udělají ze zbytků, a proto není nutné je do nářezových plánů umisťovat.

6.3. Přiřazení čísel

Protože v zakázce může být několik podobných výrobků (např. několik skříní stejného typu lišící se pouze velikostí boků nebo dveřmi) je vhodné, aby díly, které jsou pro tyto výrobky stejné, měly také stejné číslo v nářezových plánech. Z tohoto důvodu setřídím kusy nejdříve podle jména a následně podle délky. Tím dostanu stejné díly vedle sebe a můžu je projít a stejné díly označit stejným číslem. Díky tomu každý díl bude mít specifické a přesně určené číslo a stejné díly nemohou mít dvě různá.

6.4. Umístění dílů

Každý díl umisťuji přímo na plotnu ke které patří (podle návrhového vzoru Composite), který je popsán v bodě 4.3. Dokud je místo na aktuální plotně, dám díly na tuto plotnu. Pokud už je plotna plná a stále ještě potřebuji umístit nějaké díly, vytvořím novou plotnu kam zbývající díly umísťuji.

Skládání dílů na plotnu je rekurzivní funkce. Na vstupu přijde seznam dílů, které se mají ještě umístit a velikost plotny kam tyto díly umísťuji.
Začnu nalezením co nejdelšího dílu, který lze do dané velikosti plotny umístit, a umístím jej do levého horního rohu. Tento umístěný díl odeberu ze seznamu dílů, které se mají ještě umístit. Poté najdu nový co možná nejdelší díl a umístím jej pod něj, a další díl pod něj a další pod něj atd. dokud takto nevytvořím celý sloupec. Pokud již do tohoto sloupce nelze dolů přidat další díl, nastavím začátek plotny na bod, kde končí první díl a zavolám tuto funkci znovu s nově nastavenými mezemi. Tady opět tvořím co nejdelší sloupec a následně volám s novými mezemi funkci znova, dokud není celá plotna zaplněna.

```
while (true)
{
    pocetPruchodu++;
    dalsiDil = null;
    int zbyvaSirky = konec.Item2 - y;
    int zbyvaDelky = maxx - x;
    if (Nejvhodnejsi(dily, zbyvaSirky, zbyvaDelky, out dalsiDil))
    {
            dalsiDil.Umisteno = new Point(x, y);
            plotna.ObsahujeDily.Add(dalsiDil);
            dily.Remove(dalsiDil);
            y += dalsiDil.HrubaSirka;
            if (UrciMaxx) { maxx = x + dalsiDil.HrubaDelka; UrciMaxx = false; }
    }
    else
    {
        if (pocetPruchodu <= 0) { return plotna; }
        NovyZacatek = new Tuple<int, int>(maxx, zacatek.Item2);
        NovyKonec = new Tuple<int, int>(konec.Item1, konec.Item2);
        UdelejPlotnu(plotna, dily, NovyZacatek, NovyKonec);
        pocetPruchodu = -1;
    }
}
```

Obrázek 22: Ukázka kódu pro skládání dílů

Pokud ovšem nastane situace, při které by se do sloupce měl zařadit nějaký díl, který je výrazně menší než díly nad ním, vznikal by velký odpad. Proto před umístěním dalšího dílu musím zkontrolovat, zda díl nad ním má přibližně stejnou délku. V případě, že ano vše je v pořádku a mohu pokračovat. Ovšem pokud by díl byl menší, musím nastavit nový začátek umisťování na plotny a zavolat pro zbylou část plotny celou funkci znovu.

```
// pokud je o dost mensi nez predesly kus
if (staryDil != null && dalsiDil.HrubaDelka < staryDil.HrubaDelka - Config.RozdilDelek)
{
    NovyZacatek = new Tuple<int, int>(x, y);
    NovyKonec = new Tuple<int, int>(konec.Item1, konec.Item2);
    konec = new Tuple<int, int>(konec.Item1, y);
    UdelejPlotnu(plotna, dily, NovyZacatek, NovyKonec);
}
```

Obrázek 23: Ukázka kódu pro menší díl ve sloupci

Funkci hledání nejvhodnějšího dílu mám implementovanou jako funkci s výstupním parametrem. Pokud existuje díl, který se hodí do daného prostoru, funkce vrací True a nastaví výstupní parametr, v opačném případě funkce vrátí False a výstupní parametr zůstane null.

```
/// <summary> ...
private bool Nejvhodnejsi(List<Dil> dily, int zbyvaSirky, int zbyvaDelky, out Dil dil)
{
    dil = null;

    for (int i = 0; i < dily.Count; i++)
    {
        if (dily[i].HrubaSirka <= zbyvaSirky && dily[i].HrubaDelka <= zbyvaDelky)
        {
            dil = dily[i]; break;
        }
    }

    if (dil == null) { return false; }
    return true;
}
```

Obrázek 24: Ukázka kódu funkce pro hledání nejvhodnějšího dílu

6.5. Grafické znázornění

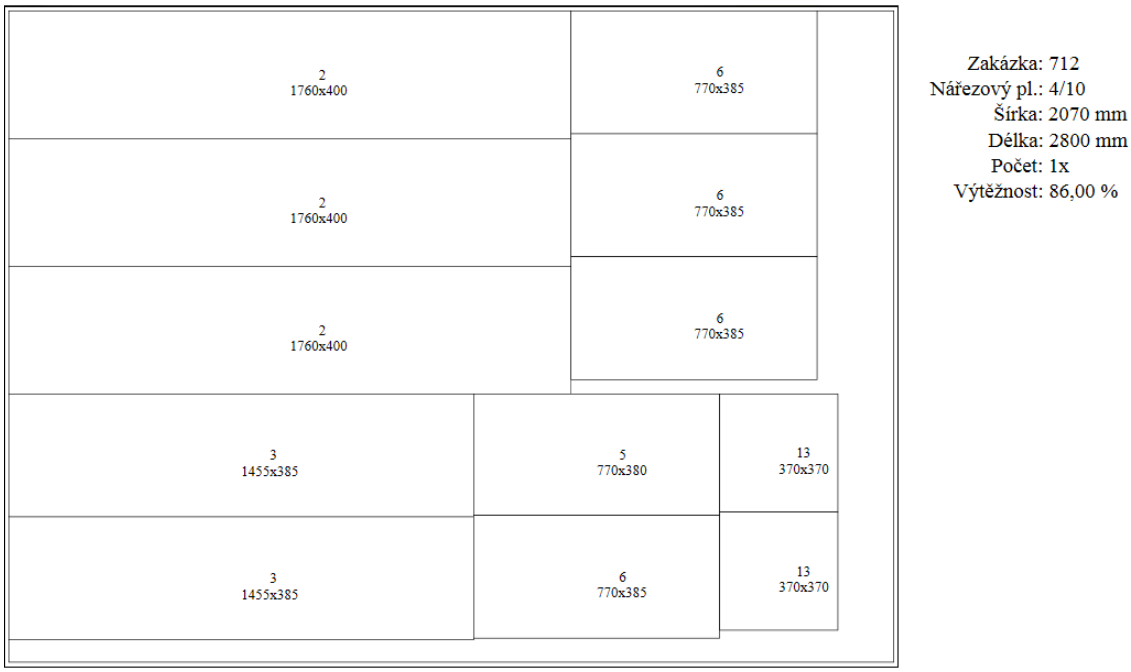
Na internetu je možno najít hned několik knihoven popř. frameworků pro práci s SVG v .NET/C#. Za zmínku stojí např. knihovna projektu http://svg.codeplex.com/ nebo si podle tutoriálu na stránkách http://www.w3schools.com/svg/default.asp vytvořit knihovnu vlastní.

Nicméně pro účely mé bakalářské práce jsem se zcela obešel bez knihovny pro práci s SVG. Používám pouze elementy Rectangle a Text a SVG soubory vytvářím pomocí textového zápisu do souboru.

Poté co jsou všechny díly umístěny na plotny mohu tyto nářezové plány vizualizovat. Pro každou unikátní plotnu vytvořím samostatný soubor. Pokud je víc ploten se stejnými díly, a tudíž i stejným rozmístěním, není nutné tyto plotny znova vykreslovat, stačí pouze u první z nich uvést počet, kolik dalších se má uřezat stejně. U každé plotny je dále uvedeno k jaké patří zakázce, číslo nářezového plánu, délka a šířka plotny a také výtěžnost této plotny.

```
<?xml version="1.0"?>
<!DOCTYPE svg PUBLIC "-//W3C//DTD SVG 1.1//EN" "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd">
<svg xmlns="http://www.w3.org/2000/svg" version="1.1" width="3600" height="2080">

<rect x="0" y="0" width="2800" height="2070" stroke="black" stroke-width="2" fill="white" />
<rect x="15" y="15" width="2770" height="2040" stroke="black" stroke-width="2" fill="white" />

<rect x="15" y="15" width="1765" height="400" stroke="black" stroke-width="2" fill="white" />
<text x="987" y="230" fill="black" font-size="45">2</text>
<text x="897" y="280" fill="black" font-size="45">1765x400</text>

<rect x="15" y="1215" width="1455" height="385" stroke="black" stroke-width="2" fill="white" />
<text x="832" y="1422" fill="black" font-size="45">3</text>
<text x="742" y="1472" fill="black" font-size="45">1455x385</text>

<rect x="1470" y="1215" width="770" height="380" stroke="black" stroke-width="2" fill="white" />
<text x="1925" y="1420" fill="black" font-size="45">5</text>
<text x="1855" y="1470" fill="black" font-size="45">770x380</text>

<rect x="2240" y="1585" width="370" height="370" stroke="black" stroke-width="2" fill="white" />
<text x="2480" y="1785" fill="black" font-size="45">13</text>
<text x="2425" y="1835" fill="black" font-size="45">370x370</text>

<rect x="1780" y="785" width="770" height="385" stroke="black" stroke-width="2" fill="white" />
<text x="2235" y="992" fill="black" font-size="45">6</text>
<text x="2165" y="1042" fill="black" font-size="45">770x385</text>

<text x="3015" y="200" fill="black" font-size="65">Zakázka: 712</text>
<text x="2900" y="280" fill="black" font-size="65">Nářezový pl.: 4/10</text>
<text x="3097" y="360" fill="black" font-size="65">Šírka: 2070 mm</text>
<text x="3081" y="440" fill="black" font-size="65">Délka: 2800 mm</text>
<text x="3090" y="520" fill="black" font-size="65">Počet: 1x</text>
<text x="2972" y="600" fill="black" font-size="65">Výtěžnost: 86,10 %</text>
</svg>
```

Obrázek 25: Část kódu vytvořeného SVG výstupu



Obrázek 26: Příklad hotového nářezového plánu

7. Měření výtěžnosti

Programoval jsem iterativně, a proto jsem měl několik verzí programu. Nad každou verzí jsem provedl několik testů a nechal vytvořit několik zakázek. Příklad rozdílů výtěžnosti budu demonstrovat na 3 zakázkách. První zakázkou je skutečná a celá zakázka, kterou jsme ve společnosti skutečně řešili, další 2 jsou pouze části zakázek, pouze ty nářezové plány, u kterých jsem čekal, že by mohly být problémy s jejich rozložením v dané verzi programu.

$$Výtěžnost = \frac{\sum_{i=1}^{n} plocha\ dílu_i\ na\ plotně}{celková\ plocha\ plotny} \cdot 100$$

Verze 0.9:

Zakázka 712	
Výtěžnost v %	Počet stejných ploten
85,35	1
60,11	2
60,43	1
60,90	1
55,87	1
48,32	2
29,92	1
25,24	3
25,44	1
25,57	2
25,44	1
24,88	1
24,41	9
24,92	1
21,48	1
2,36	1
∅: 34,30	Celkem: 30

Zakázka 726a	
Výtěžnost v %	Počet stejných ploten
83,73	1
7,71	1
∅: 45,75	Celkem: 2

Zakázka 726b	
Výtěžnost v %	Počet stejných ploten
57,38	1
27,36	1
∅: 42,36	Celkem: 2

V této verzi byl problém se špatným určováním konce sloupce. To fungovalo pouze u první tabule a na dalších již ne. Také zde nebylo zohledněno skládání výrazně menších dílů ve sloupci.

Verze 1.0

Zakázka 712	
Výtěžnost v %	Počet stejných ploten
85,35	3
85,67	1
86,14	1
81,37	1
78,62	1
73,83	1
74,02	1
73,44	1
73,25	2
73,34	1
∅: 79,16	Celkem: 13

Zakázka 726a	
Výtěžnost v %	Počet stejných ploten
66,31	1
25,18	1
∅: 45,75	Celkem: 2

Zakázka 726b	
Výtěžnost v %	Počet stejných ploten
64,19	1
20,54	1
∅: 42,36	Celkem: 2

Ve verzi 1.0 se oproti předchozí verzi značně zvýšila výtěžnost a díky tomu i snížil počet ploten na celou zakázku. Problém ovšem opět nastával při menších dílech ve sloupci.

Verze 1.1

Zakázka 712	
Výtěžnost v %	Počet stejných ploten
85,35	3
85,67	1
86,14	1
86,10	1
73,89	1
73,83	1
74,02	1
73,44	1
73,25	2
73,34	1
∅: 79,16	Celkem: 13

Zakázka 726a	
Výtěžnost v %	Počet stejných ploten
91,51	1
∅: 91,51	Celkem: 1

Zakázka 726b	
Výtěžnost v %	Počet stejných ploten
84,73	1
∅: 84,73	Celkem: 1

Verze 1.1 se oproti verzi 1.0 poměrně liší. Při přidání podmínek pro případy, kdy by se do sloupce měl přidat díl, který je výrazně menší, než předcházející, se stal kód značně složitým a hledat v něm chyby bylo velmi obtížné. Z tohoto důvodu jsem celou funkci pro umisťování dílů

přepracoval do podoby rekurzivní funkce, která najednou nezpracovává celou plotnu, ale vždy pouze její část. Tím jsem vyřešil i problém menších dílů, které jsou již odděleny od původního sloupce a zpracovávají se samostatně.
Tato verze již je optimální a zároveň zachovává možnost řezat tyto nářezové plány na pile s předřezem.

8. Závěr

V této bakalářské práci jsem se zabýval optimalizací nářezových plánů. Tyto nářezové plány byly tvořeny pro pily s předřezovými kotouči z čehož vyplývají jistá omezení při jejich tvorbě.

Vytvořil jsem také systém pro tvorbu a editaci zakázek, z nichž se tyto nářezové plány vytvářejí. Tyto zakázky ukládám do XML souborů a výsledné nářezové plány vykresluji pomocí SVG formátu, který je XML formátu velmi blízký.

Nářezové plány se mi podařilo optimalizovat i přesto, že literatura pro tento typ nářezových plánů je málo dostupná. Musel jsem tudíž vycházet z již vytvořených nářezových plánů a odhadovat algoritmické postupy, které byly při jejich výrobě použity.

Do budoucna bych chtěl z tohoto projektu udělat WCF službu a nářezové plány vytvářet pomocí evolučních algoritmů. Také bych chtěl tento projekt rozšířit o zpracování plošných materiálů s tzv. „uni“ strukturou. To znamená, že plotna má jednolitý vzor a u jednotlivých dílů je možné měnit délku za šířku a naopak.

Při tvorbě tohoto projektu jsem využil znalosti nasbírané v předmětech, které se zabývají programováním, a tyto znalosti jsem si zdokonalil a dále rozšířil. Nabyté zkušenosti jistě využiji v dalších projektech.

9. Reference

[1] *DřevoTrust.cz.* [Online] [Citace: 03. 04 2013.] http://drevotrust.cz/cz/rubriky/produkty/plosny-material/drevotriskove-desky-dtd/.

[2] edison.sso.vsb.cz. [Online] [Citace: 17. 04 2013.] https://edison.sso.vsb.cz/wps/myportal/student/!ut/p/b1/.

[3] Merick Calk WF. *www.softconsult.cz.* [Online] [Citace: 13. 04 2013.] http://www.softconsult.cz/cz/mcwf/index.htm.

[4] OPTIMIK. *www.cad.cz.* [Online] [Citace: 13. 04 2013.] http://www.cad.cz/component/content/article/2604.html.

[5] Nowy Rozkroj. *cz.pro100.eu.* [Online] [Citace: 13. 04 2013.] http://cz.pro100.eu/nowy-rozkroj.

[6] XML Introduction - What is XML. *www.w3schools.com.* [Online] [Citace: 22. 04 2013.] http://www.w3schools.com/xml/xml_whatis.asp.

[7] SVG Intro. *www.w3schools.com.* [Online] [Citace: 22. 04 2013.] http://www.w3schools.com/svg/svg_intro.asp.

[8] Composite Design Pattern in C# and VB.NET. *www.dofactory.com.* [Online] [Citace: 25. 04 2013.] http://www.dofactory.com/Patterns/PatternComposite.aspx.

[9] Operační systémy a programování - Operacni_systemy_a_programovani.pdf. *www.elearn.vsb.cz.* [Online] 30. 06 2008. [Citace: 24. 04 2013.] http://www.elearn.vsb.cz/archivcd/FS/OSP/Operacni_systemy_a_programovani.pdf.

[10] **Tanenbaum, Andrew S.** *Modern operating systems.* New Jersey : Prentice-Hall, 2001. ISBN 0-13-031358-0.

10. Obsah DVD

Následující tabulka popisuje umístění souborů na DVD a jejich popis.

/doc	Text bakalářské práce ve formátu PDF/A
/src/BP	Microsoft Visual Studio Solution
/src/BP/ Kusovnik	Testovací výrobky
/src/BP/ Zakazky	Testovací zakázky + jejich nářezové plány